北京科技大学
University of Science and Technology Beijing

密级：公开

本科生毕业设计(论文)

题 目：基于流量监听的被动式漏洞扫描器设计与实现

作 者：蒲应元

学 号：41524529

学 院：计算机与通信工程学院

专 业：信息安全

成 绩：

2019 年 05 月

本科生毕业设计(论文)

题　　目: 基于流量监听的被动式漏洞扫描器设计与实现

英文题目: Design and Implementation of Passive Vulnerability Scanner Based on Traffic Monitoring

学　　院: 计算机与通信工程学院

班　　级: 信息安全 1501

学　　生: 蒲应元

学　　号: 41524529

指导教师: 王志明　　**职称:** 副教授

指导教师:　_____职称：_____

声　　明

本人郑重声明：所呈交的论文是本人在指导教师的指导下进行的研究工作及取得研究结果。论文在引用他人已经发表或撰写的研究成果时，已经作了明确的标识；除此之外，论文中不包括其他人已经发表或撰写的研究成果，均为独立完成。其它同志对本文所做的任何贡献均已在论文中做了明确的说明并表达了谢意。

学生签名：_____　　_____年___月___日

导师签名：_____　　_____年___月___日

毕 业 设 计（论 文）任 务 书

一、学生姓名：蒲应元　　学号：41524529

二、题目：基于流量监听的被动式漏洞扫描器设计与实现

三、题目来源：真实 ☐ 、　　自拟 ☑

四、结业方式：设计 ☑ 、　　论文 ☐

五、主要内容：

1、设计一个基于 B/S 架构的扫描器，能够对于流量进行被动分析，并在浏览器前端显示结果。

2、自行设计和配置相关漏洞扫描规则，实现在流量数据包层面对常见漏洞的检出，如 XSS 和 SQL 注入等。

3、扫描器基本优化，包括多线程、分类、前端 UI 等。

六、主要（技术）要求：

1、使用 python 的 web 框架 tornado，完成基本 B/S 架构，并达到可以抓取流量数据包并分析显示的效果。

2、设计相关漏洞（sql、xss 等）的扫描规则，能够从所监听的请求流量中实现 5 种以上漏洞的检出。

七、日程安排：

第 1 周：查阅资料，确定毕业设计题目，明确毕业设计的流程；

第 2-4 周：查阅国内外相关资料，熟悉 tornado 框架，撰写开题报告；

第 5-7 周：设计扫描器系统的实现方案，并搭建基本 web 环境，实现流量的代理和抓取，完成翻译工作；

第 8-10 周：设计相关漏洞的扫描规则，并进行编码调试，撰写并提交中期检查表，准备中期答辩；

第 11-12 周：完善扫描器前端界面，并优化扫描器性能；

第 13-15 周：进行相关测试，总结实验结果并撰写毕业论文，准备答辩。

八、主要参考文献和书目：

[1] 刘明明. Web 应用漏洞扫描器的设计与实现[D]. 电子科技大学, 2014.

[2] Begum A, Hassan M M, Bhuiyan T, et al. RFI and SQLi based local file inclusion vulnerabilities in web applications of Bangladesh[C]. International Workshop on Computational Intelligence, 2017.

[3] Al-Khurafi O B, Al-Ahmad M A. Survey of Web Application Vulnerability Attacks[C]. International Conference on Advanced Computer Science Applications & Technologies, 2016.

[4] 陈翔. Web 网站扫描器的设计与实现[J]. 信息与电脑(理论版), 2016(8): 99-100.

[5] 李威, 李晓红. Web 应用存储型 XSS 漏洞检测方法及实现[J]. 计算机应用与软件, 2016, 33(1): 24-27.

[6] 基于 B/S 架构漏洞扫描技术的研究与实现[D]. 北京邮电大学, 2015.

[7] 基于数据包分析的被动漏洞扫描技术[D]. 西安电子科技大学, 2015.

指导教师签字：　　年　　月　　日

学 生 签 字：　　年　　月　　日

系（所）负责人章：　　年　　月　　日

摘　要

信息化的时代中，异彩纷呈的的网络技术虽然得到了蓬勃发展，但于此同时网络空间的安全形势显得不容乐观。各式各样的 Web 应用服务挂载在互联网上，这也同时说明有着各式各样的潜在地安全风险存在于这些应用程序中。网络安全的问题刻不容缓，如今的攻击者利用自动化攻击手段，更加快速的发现网站服务的漏洞，从而盗取用户信息，危及个人隐私。而与之做对抗的安全测试人员，也应有自动化的测试工具，在程序正式上线之前进行更好的安全测试，更早的发现潜在的漏洞和风险，并及时修复。

本文主要针对一种被动式的漏洞扫描手段展开研究，设计并实现了一个半自动化的基于流量监听的被动式漏洞扫描器。主要通过对数据包的抓取分析，使用特征匹配的方法来发现漏洞。具体的研究内容如下：

（1） 调研并分析了现已有的 Web 应用安全漏洞扫描技术，对比主动式和被动式的扫描技术，分析其异同和优劣。

（2） 设计一套被动式漏洞扫描器的扫描流程和异步响应的工作队列机制，使得扫描器有更快的响应速度和更高扫描效率。并且通过 B/S 架构设计 Web 端扫描器，使扫描器突破开发环境的限制，随时随地进行快速的安全测试和扫描。

（3） 通过收集和归纳不同类型漏洞的特征匹配规则，形成对应于特定类型漏洞规则文件，来提高匹配精度，必要时可通过修改扫描器设置，来对单一类型漏洞进行更广泛和更高效的扫描测试。

被动式漏洞扫描技术有着更好的扫描精度，在高水平的测试人员使用下，可以成倍提高安全测试的效率，并节省相应的资源成本，更精准快速的完成漏洞探测，发现安全问题。相比于主动式扫描，在发现更深层次的安全问题方面有着独特的优势。

关键词： 扫描器，被动式扫描，网络安全，Web 应用安全，流量监听

Abstract

In the era of informationization, although the colorful network technology has been booming, at the same time, the security situation of cyberspace is not optimistic. A wide variety of web application services are mounted on the Internet, which also means that there are a variety of potential security risks in these applications. The issue of network security is an urgent task. Today's attackers use automated attacks to more quickly discover vulnerabilities in website services, thereby stealing user information and endangering personal privacy. The security testers who are against it should also have automated test tools to perform better security tests before the program is officially launched, to identify potential vulnerabilities and risks earlier, and to fix them in time.

This paper focuses on a passive vulnerability scanning method, and designs and implements a semi-automatic passive vulnerability scanner based on traffic monitoring. The vulnerability is discovered by using feature matching methods mainly through crawling analysis of data packets. The specific research contents are as follows:

(1) Investigate and analyze the existing web application security vulnerability scanning technology, compare active and passive scanning technologies, and analyze their similarities and differences and advantages and disadvantages.

(2) Design a set of passive vulnerability scanner scanning process and asynchronous response work queue mechanism, so that the scanner has faster response speed and higher scanning efficiency. And through the B / S architecture to design a Web-side scanner, the scanner to break through the limitations of the development environment, fast security testing and scanning anytime, anywhere.

(3) Improve the matching accuracy by collecting and summarizing the feature matching rules of different types of vulnerabilities to improve the matching accuracy. If necessary, modify the scanner settings to make a single type of vulnerability more extensive and efficient. Scan test.

The passive vulnerability scanning technology has better scanning accuracy. Under the use of high-level testers, it can double the efficiency of safety testing, save the corresponding resource cost, complete the vulnerability detection more accurately and quickly, and find security problems. Compared to active scanning, it has unique advantages in discovering deeper security issues.

Key Words： Scanner，Passive scanning，network security，Web Application，Traffic monitoring

目　录

插图清单

1 引　言

1.1 课题背景

近些年来，各类网络技术发展迅速，web 应用服务的类型也与日俱增，随之而来的安全问题也越来越得到重视。2017 年 6 月 1 日，我国第一部《网络安全法》正式实施，我国网络安全也迈入了法治新阶段。然而在 2018 年，实际中对于网站 Web 服务的整体攻击的指数仍然是不断上升的趋势。从知道创宇团队的统计数据来看，针对于 Web 应用的攻击平均每日攻击量高达 8 亿多次，同时存在较大浮动，会在某个时间节点产生巨大流量的攻击，如在 2018 年的 8 月份出现了攻击流量的高峰期，峰值高达单日 49 亿余次。2018 年的整体网络攻击趋势如图 1-1。

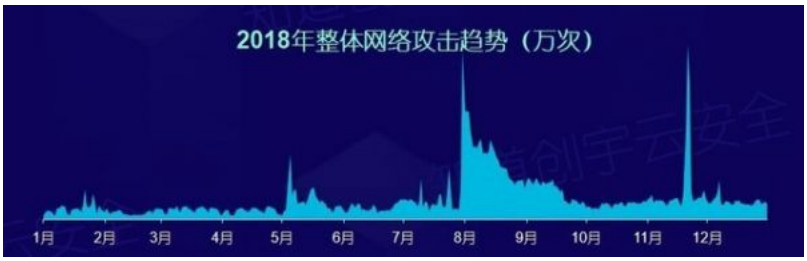

图 1-1 2018 年整体网络攻击趋势（知道创宇报告）

而在如此庞大的攻击数量之下，自动化的攻击手段已经成为必然，带有攻击性的扫描器在 Web 攻击手段中占比高达 61%。自动化的攻击手段使得攻击的数量愈多，持续的时间愈久，发现漏洞的速度也愈快，这就使得网络安全的保障更加困难[1]，在图 1-2 中可以看出，各种 Web 应用攻击手段不同的占比，很明显的看出扫描器占比最大。

图 1-2 2018 年 Web 攻击手段分析（知道创宇报告）

与此同时，各类的安全技术也在快速的发展和成熟，在这期间诞生了一系列的安全防护措施，如防火墙、WAF、入侵检测系统（IDS）和入侵防护系统（IPS）等待。这些技术可以发现传统的攻击特征，并进行拦截和过滤。但这也只是一些有效的防护手段，并不是最终的解决方案，它们还需要进一步的的完善和升级。攻击者的攻击手段总是与日俱新的，想要更好的测试网站 Web 服务的安全性，就得改进扫描检测的技术，现在已有的一些 Web 漏洞扫描器，如 APPSCAN、AWVS 等都是基于其中的爬虫模块，主动的向 Web 服务发送载荷（payload），在对一些大型服务测试时，为了其快速广泛的扫描，其扫描过程并发高且容易被识别为攻击性强，流量特征明显，从而触发防火墙等安全系统的保护措施而被阻断[2] [3] 。

1.2 研究意义

网络安全问题防不胜防，攻击者们利用自动化的攻击手段，更加快速的发现网站服务的漏洞，盗取用户信息，危害个人隐私。而有效的解决手段就是及时的发现漏洞，修补漏洞，让攻击者无可利用之处。以彼之矛，攻彼之盾。从安全测试的方面来看，自动化漏洞的扫描技术可以更迅速的发现自己的弱点，能更好的在产生威胁之前发现漏洞，并修复漏洞。

安全测试中的攻与防是一个相互促进的结果，这些年来随着防护技术的进步，现已有的一些入侵检测系统、入侵防护系统已经做到了较为成熟的地步，一些传统的攻击方式、扫描手段已经可以被有效的发现和过滤。自动化检测技术的进步，对于安全测试人员改进和更新已有的测试方式，并用更加有效的手段来发现存在的安全隐患具有重大意义[4] 。

对于现有的扫描器来说，其扫描方式一般有主动扫描和被动扫描两种。现在市面上各种付费商用的或者免费开源的扫描器，大多都是以主动扫描的方式来进行，而主动扫描一般用于黑盒测试，形式一般为提供一个带扫描扫描目标服务或者目标网络，然后由扫描器中的爬虫模块递归的爬取 Web 网站服务中的所有跳转链接，对于抓取到的数据中的参数进行修改变形后提交，并进行重放测试，通过数据包中返回的报文状态码、数据大小、数据报正文页面特征、数据内容和关键字等特征进行判断，判别该请求中是否含有对应的漏洞存在。其过程一般为全自动化的测试，开始由测试人员设置测试目标，启动之后则自动进行，直至生产测试结果。而与之相对的，另一种扫描方式则为被动式的扫描，与主动式扫描相比，被动式扫描并不直接向所测试的服务器发出请求[5] ，也不进行大规模的爬取行为，而是通过抓取数据包，监听分析测试人员的请求流量，来进行漏洞的发现，同时这样也不会影响实际网

络的运行情况。这种扫描方式则更加偏向半自动化，整个扫描过程中需要测试人员的配合，依赖于测试人员的渗透测试水平，但是有着更强的目的性和针对性，可以发现系统潜在的更深层次的风险。

从扫描器的数据来源方面来看，主动扫描的数据源都是来自于其爬虫模块对网页的爬去结果，对于一些独立的页面或者 API 接口就无法覆盖到，可能会存在漏检的情况，而且对于一个大型的站点来说，其页面的链接丰富多样，网站结构层次也很有深度，如果使用主动扫描，对于爬取设置的广度和深度不同，往往获得结果也是有所不同的，这其中所使用的时间，资源消耗也非常值得考量。如果爬取时并发量过高，很有可能导致正常的使用环境受到影响，而较低的并发量，对于庞大的爬取任务又是一个挑战。而被动扫描在数据源获取这个环节就大有不同，他的数据直接来源于测试人员的访问流量，测试人员有针对性的进行测试探察，获取的流量也具有针对性，因而也可更加快速精确的发现漏洞。主动的扫描的精确度和效率都无法与被动扫描相比拟。因此对于被动扫描技术的研究深有意义[6] 。

1.3 研究方法及研究目标

本文旨在于设计实现一个基于流量监听的被动式漏洞扫描器，能够帮助安全测试人员在渗透测试中更好的发现系统潜在风险。通过调研、设计、实现、测试、优化改进几个流程来完成最终的设计与实现。

主要的研究从以下几点进行：

（1） 通过深入的背景调查和形势分析，对当前的网络安全形势和已有的漏洞扫描解决方案做了详细的阐述。

（2） 通过调查研究已有的漏洞扫描器，熟悉其工作原理和设计结构，对不同扫描器的漏洞扫描方案进行比较分析，并收集整理其有效的扫描规则。

（3） 通过总结已有的漏洞扫描解决方案，提出被动式漏洞扫描方案，设计出漏洞扫描器基本工作流程，并通过编程实现。

（4） 通过搭建含有漏洞的测试环境，对本文中实现的被动式漏洞扫描器进行测试，根据测试结果优化扫描器性能和扫描规则。

1.4 论文组织结构

本文的结构可以分为以下几个部分：

第一章是引言部分，主要论述了课题背景、研究意义、研究方法及目标

和论文组织结构几个部分，是一个对于课题的初步介绍和调查，由此展开整片文章的论述。

第二章是对现有的漏洞扫描技术的一个综合调研，从针对于不同层面的扫描目的来分为两个部分说明，分别是针对主机层和Web层的漏洞扫描技术，而在第二部分中也引出了本文的核心，被动式的漏洞扫描方案。第三部分是对于本文涉及到的扫描器实现过程出现的漏洞的综述。

第三章是基于上一章充分的调查论证和分析，提出了被动式漏洞扫描器的详细需求分析，然后通过明确需求制定了扫描器的整体设计方案，主要从扫描器的整体工作流程，扫描器的架构设计，以及漏洞规则匹配的设计三个方面来阐述了被动式漏洞扫描器的设计思路。

第四章是对于扫描器的功能的最终实现，也是本文的核心部分，分别从六个不同的核心功能点出发，充分阐述了扫描器的实现过程和实现方法。这同时也是扫描器编程实现的过程，上一章节中明确的架构设计、整体设计和核心的逻辑思路也是在这一部分进行了实现。

第五章是扫描器系统的测试验证和分析，对编程实现完成的扫描器，还需通过漏洞环境的测试来确定其可用性，以及使用的友好性和方便性。同时也是对扫描器的具体使用过程做一个说明。

第六章是本文结论，主要对调研到设计再到实现和测试的工作总结，也是对整个课题的一个总结。第二部分还提出了对于该被动式扫描方案的未来展望和预期。

2 漏洞扫描技术分析

漏洞也称作脆弱性（Vulnerability），是指在计算机系统中因为使用一些程序、协议、配置而导致存在的一些缺陷，有的甚至就是操作系统本身存在的一些缺陷，这些缺陷的存在就导致了系统会被黑客攻击的风险。这种风险小到泄漏一些系统版本信息，大到攻击者可直接攻入服务器，拿到服务器的高权限然后任由攻击者操作，由此会引发的安全问题远超我们的想象。

根据漏洞扫描技术的定义，漏洞扫描技术是指基于漏洞数据库或者通过扫描等手段对指定的远程或者本地计算机系统的安全脆弱性进行检测，从而发现可利用漏洞的一种安全检测（渗透攻击）行为。但这种基于漏洞数据库的扫描技术有着一定的局限性，只能确保一些比较固定的安全问题不会发生，如某个 CMS 存在已经纰漏过的 CVE 或者版本漏洞，但是无法挖掘出一个系统更加深入的安全问题。

现如今的漏洞扫描技术已经不单单是自动而广泛化的扫描了。无论是企业还是政府对于安全的要求越来越高，基础安全的保障做的越来越到位，这时对于深层次的安全问题挖掘也显得重要起来。高水平的渗透测试人员配合半自动化的扫描技术也变得越来越受欢迎。

漏洞扫描技术从不同的实现角度可分为针对主机层面的漏洞扫描技术和针对 Web 应用层面的漏洞扫描技术。而后者又更具扫描方式不同可分为被动式漏洞扫描技术和主动式漏洞扫描技术。

2.1 针对主机层面的漏洞扫描技术

2.1.1 主机漏洞扫描技术的原理

针对主机层面的漏洞扫描主要的关心对象还是被扫描网络中的资产和这些资产可能潜在的风险，目标则是指定单独的扫描对象或者一个网络，同时扫描器也会对这些资产设备上运行的一些系统和程序进行一个简单的探测。

一次完整的目标网络主机扫描大致可分为三个阶段：

第一阶段，设定目标并进行初步的主机和网络发现，确定目标网络下存在的实体或者虚拟资产的数目。

第二阶段，发现这些目标之后进行进一步的信息收集，包括操作系统类型和版本，运行的服务及服务的版本，对一一个目标网络的扫描还包括对于网络的拓扑结构的探测与路由信息的收集。

第三阶段，根据这些收集到的信息，在漏洞库中进行匹配，是否有已知漏洞的存在被发现，是否一些已知版本号的服务存在安全漏洞等。

2.1.2 主机漏洞扫描的常用方法

ICMP 主机发现，用于发现目标网络中存活主机的最快方式，也就是最常见的 ping 命令的使用，但是不同的网络系统中由于网络管理员的设置不同，对于主机发现的要求也就五花八门，比如有的网络中防火墙规则禁用 ICMP 包的传输，此时就须使用其他方式来进行主机探测。在这其中最出名的就是 Nmap 扫描工具，它提供了 TCP SYN/ACK，UDP 和 ICMP 灵活组合的使用，甚至有一些绕过防火墙、欺骗防火墙的扫描策略，使得主机发现的工作更加简单也更加全面。这一方法使用在扫描的第一个阶段。

操作系统的探测即操作系统指纹识别，这一方法主要是用于识别某台物理机上运行的操作系统的类型。通过分析主机系统向网络发送的数据报文中包含的某些协议特征、参数和默认配置，借此来推断出发送这些数据包的操作系统类型，而这些数据特征我们则称为系统指纹。一般也可以使用 Nmap 来进行操作系统的发现,这方法使用在扫描的第二个阶段。

端口扫描技术，服务器上开发的一个个端口就是一个个通信的通道，最常见的 22 端口就是用来远程 ssh 连接服务器的。不同的端口运行着不同的服务。通过对端口的扫描可以知道服务器上运行着那些服务。常见的端口扫描方式有 TCP connect、TCP SYN、TCP FIN 扫描等等。

常用的针对主机层面的漏洞扫描器有 Nessus 和 Nmap。

2.2 针对 Web 应用层面的漏洞扫描技术

Web 安全漏洞扫描技术指的是，安全检测技术在 Web 应用程序漏洞检测方面的应用，通过分析 Web 应用程序相关漏洞的形成原因和攻击原理，发展而来的扫描检测技术。通过对不同类型漏洞的深入分析，分别研究他们在不同 Web 服务下的表现形式，提取漏洞反映出的共有的特征，形成一类特征码，然后再使用自动化爬虫和部署技术在此基础之上，研发出针对这些漏洞的检测工具。在对测试目标扫描前，提前预置扫描时需要运行的各种参数和对于漏洞的检测规则或者特征匹配规则，然后启动扫描器，一旦扫描器匹配到相关漏洞特定的特征或者页面信息，就将该漏洞的详细信息都保存下来，并在扫描完成之后形成扫描报告。

随着网络技术和攻击手段的不断发展，漏洞扫描技术也在不断的发展，

从第一个漏洞扫描器 WarDialer 开始，扫描技术的自动化程度就在不断增加。WarDialer 可以结合使用几种已知的扫描技术实现自动扫描，还可以按照统一的格式将扫描结果记录下来[7] 。在这种自动化技术的发展趋势之下，漏洞扫描技术也有了一个长足的进步，Web 应用程序的安全性能也得到了迅速的提升。早在安全技术刚刚兴起的时候，黑客们凭借简单的手段即可威胁到网站的安全，多如网站挂黑页的手段在那个年代频频出现，但现在随摄安全性的提升和相关法律法规的完善，如此状况已经很少见了。随着日益复杂和多样化的 Web 应用服务的出现，使得一些安全问题和漏洞的出现成为不可避免的事，另一方面也正是这样的局势下促进了安全扫描技术的再次发展[8] 。

漏洞扫描技术一般通过对应用服务的扫描，来发现一些潜在的洞或者风险，而且现今的扫描器在扫描的同时也同时聚合了多种信息探测搜集的功能，如主机扫描、端口探测、代码审计等功能。不仅仅提供了发现安全隐患的功能，更为安全测试人员提供了一个信息收集、风险评估、分析系统的便捷信息来源。

2.2.1 基于爬虫的主动式漏洞扫描技术

作为一个基于网络（Web 层面）的漏洞扫描器，一般都是从外部攻击者的角度来检测系统中是否存在一些安全漏洞[9] 。所以扫描器大多也都是基于爬虫模块的开发。一般的基于爬虫的主动式漏洞扫描器的工作流程分为三个阶段：页面爬取、站点探测和漏洞发现[10] 。

一般来说，网页爬取模块和漏洞发现模块都是基于 HTTP/HTTPS 协议来实现的。Web 站点静态资源和源代码的抓取、站点拓扑结构的绘制以及漏洞的发现检测，都是通过发送 HTTP/HTTPS 请求，并且分析相应的请求数据包响应头部、响应内容和响应状态码来实现的。网络爬虫模块作为扫描器的重要组成部分，他会从一个初始的目标集开始来对整个网站进行爬取，下载所有的网页内容，将网页中检测出的其他同源网页跳转链接再次添加到待爬取的队列中，反复迭代爬取所有的站点链接，直至到达指定的爬取深度或者全部爬取完为止。不同的扫描器对于爬虫有不同的设计和优化，主要的爬取策略还是分为深度优先遍历策略、广度优先遍历策略和最佳优先遍历策略这三种。

深度优先遍历策略是指，爬虫会从起始点开始，一个链接接着一个链接跟踪下去，直至处理完这个链接的所有跳转之后再从下一个目标集合中的对象开始。广度优先搜索策略是指，在抓取过程中，先爬取任务队列中的所有

目标，对于在这些目标爬取过程中检测出的新链接并不立刻开始爬取，而是先把他们依次加入待爬取的队列，等当前所有的任务完成后，在从队列中取出新加入的链接进行爬取，反复循环直至结束。而最佳优先遍历策略是指，按照预先制定好的网页相似度算法，评判接下来待扫描的网页和当前网页的相似程度或者一些特征词主题的相关程度，选择一个评价最高的网页进行下一步爬取，这样的结果就是，它会遵循算法的原则来选择爬去目标，爬取的内容的质量也很大程度上取决于算法的性能和适用性，有可能会忽略一些存在漏洞的页面，使得扫描器的精确度下降。随着爬虫技术的不断发展，现在的爬虫定制化程度更高，更加针对所需要进行漏洞扫描的对象来选择合适的爬取策略[11] 。

扫描器中的站点探测和漏洞发现模块是扫描器的核心，当爬虫爬取到足够的信息之后，一般会将这些信息存入数据库，然后由探测模块来和数据库发生关联，进行漏洞扫描原理的实现，其核心还是基于特征码匹配的原理来发现一些已知的漏洞。一个网站所有页面的数据量可能非常庞大，探测模块需要对所有的请求数据进行全面而详尽的分析探测，通过遍历数据库中的所有数据，找到这些动态交互点中可以向 Web 提交的参数类型，然后由漏洞检测模块通过对请求参数的变形向 Web 页面重放请求，再通过响应信息的不同和特征码的比对从而发现漏洞。对于一般的 Web 服务来说，可以能的探测点有 URL 路径、GET 方法中的参数、POST 方法请求体中的参数、请求 HEADER 中的字段、COOKIE 中的键值，响应体等[19] 。

2.2.2 基于流量监听的被动式漏洞扫描技术

被动式的漏洞扫描技术也是相对于主动扫描来讲的，不同于主动式的漏洞扫描，它的数据来源并不是对于页面的爬取，而且通过对于测试人员流量的监听截取来获得的。同样的，被动式漏洞扫描技术按照功能同样也可以分为数据来源、数据处理方式和漏洞检测等不同部分。

对于被动式扫描的数据来源主要有两种类型，一种是通过加载旁路设备直接镜像所有通过交换设备流量，另一种则是通过代理服务器的方式，主动将测试流量发向代理服务器[12] 。在第一种方法中，需要部署一台额外的设备在核心交换机上，以便在不影响正常使用的情况下抓取所有流量，这种方式虽然花销较大，但是性能很好，可以做到抓取全协议流量，抓取精度和范围都更佳。对于这些流量 中的 HTTPS 协议的流量，需要证书和密钥揭秘才可以按照正常的 HTTP 数据包来解析。

在第二种基于 HTTP 代理的方式中，只需要配置代理服务器，测试人员使用代理进行正常的测试访问，代理服务器上运行着扫描器，扫描器会接收所有发向指定端口的流量，然后在扫描器内部编程来实现数据包解析、去重和统计处理，解析结果发送到任务队列进行下一步的漏洞扫描[13] 。

对于 HTTP 的流量解析，一般都是按照元组的方式来重组数据包，读出每一个报文中标记的请求方式、请求地址、请求的参数等，将他们提取为 JSON 格式的数据，来建立任务，进行下一步工作。

被动式漏洞扫描技术的漏洞检测模块其原理也和主动式的相近，都是基于漏洞的特征码匹配。只不过通过流量监听收集来的数据更加具有针对性，测试者可以有倾向的访问一系列可能存在某一种漏洞的链接，从而达到迅速精确的检测效果。由于数据的来源可控，针对一些基于爬虫的扫描器无法获取的数据，如 API，和一些会发生意外跳转和关闭的页面有更好的效果，现在的各类网站都有反爬虫混淆的功能，这对于主动式扫描器的爬虫技术也是一个挑战。

2.3 Web 安全漏洞综述

随着越来越多的服务移植到 Web 的平台上，以及 Web2.0 时代的快速发展，针对 Web 的恶意攻击行为成为当前网络安全最为严重的问题。OWASP（Open Web Application Security Project） 2017 TOP 10 安全报告给出了目前广泛的十大应用安全风险漏洞如图 2-1 所示[14] ：

2013年版《OWASP Top 10》		2017年版《OWASP Top 10》
A1 – 注入	→	A1:2017 – 注入
A2 – 失效的身份认证和会话管理	→	A2:2017 –失效的身份认证
A3 – 跨站脚本（XSS）	↘	A3:2017 –敏感信息泄漏
A4 – 不安全的直接对象引用 [与A7合并]	∪	A4:2017 – XML外部实体（XXE） [新]
A5 – 安全配置错误	↘	A5:2017 – 失效的访问控制 [合并]
A6 – 敏感信息泄漏	↗	A6:2017 – 安全配置错误
A7 – 功能级访问控制缺失 [与A4合并]	∪	A7:2017 – 跨站脚本（XSS）
A8 – 跨站请求伪造 （CSRF）	☒	A8:2017 – 不安全的反序列化 [新，来自于社区]
A9 – 使用含有已知漏洞的组件	→	A9:2017 –使用含有已知漏洞的组件
A10 – 未验证的重定向和转发	☒	A10:2017 – 不足的日志记录和监控 [新，来自于社区]

图 2-1 OWASP TOP 10 2017 报告

从图中可以看出，注入漏洞的危害一直高居不下，且新出现了 XML 外部实体漏洞，说明在 Web 表示形式的快速变更下，XML 的安全问题也愈发凸显出来。

本节主要详细罗列出几类在本扫描器实现中涉及到的漏洞，通过描述漏洞的成因和检测方法，来展开接下来扫描器设计的工作。这些漏洞主要分为三大类，SQL 注入型漏洞、XPATH 注入漏洞、XSS 漏洞。其中 SQL 注入漏洞又可细分为报错型 SQL 注入、布尔型 SQL 注入和时间型 SQL 注入三种形式。

2.3.1 布尔型 SQL 注入漏洞

从 OWASP 的报告我们可以看出，SQL 注入漏洞是 Web 层面里面最危的漏洞之一。SQL 注入漏洞的核心就在于非预期指令在数据库的执行，Web 网页的交互避免不了的需要把用户的输入转换成 SQL 语句，然后在数据库中查询到结果并返回，但是当用户有意的错误输入，如在输入中故意构造出引号闭合，就有可能让一些用户输入数据变成危险的 SQL 指令从而被执行[14] 。

而布尔型的 SQL 注入漏洞是指，命令的执行与否会返回不同的页面状态，通过比较返回数据包的异同和页面相似度即可发现是否存在该类型漏洞，这也是最基础的 SQL 注入漏洞类型，我们可以通过请求相同的输入指令再用 AND 条件加以不同的正确或者错误的表达式来判断，是否有指令在数据库成功执行[15] 。

2.3.2 报错型 SQL 注入漏洞

报错型 SQL 注入漏洞的形成原因大多是因为在查询语句中使用了一些特殊的函数导致的 SQL 查询出错，例如 floor(rand()*2)、updatexml、exp()等等，这些都是 SQL 语法中比较特殊的函数，在恶意构造的情况下就可能触发查询错误，同时服务器会接收到错误信息返回在页面上，而在返回的过程中则会读取 SQL 语句中的其他输入，如果攻击者使用 CONCAT 函数连接了包括报错部分的多个查询条件，则其他部分的 SQL 语句得到执行，且数据会被带回前端浏览页面，从而造成了报错型 SQL 注入漏洞。

对于这种漏洞的检测方式也较为简单，在刻意构造的 payload 请求中加入含有报错查询语句的参数，然后在服务器返回的数据包中进行匹配，如果发现了特定的报错信息，则视为存在报错型 SQL 注入漏洞[17] 。

2.3.3 时间型 SQL 注入漏洞

时间型的 SQL 注入漏洞，其核心和其他 SQL 注入漏洞也相同，同样是是非预期 SQL 命令得以执行，但是这一类漏洞更加适用于描述那些关闭了网站错误显示，无法单从网页的变化看出 SQL 命令是否执行，这时候使用 SQL 语法中的 Sleep 函数就可以判断出命令的执行，如果 Sleep 函数在后台数据库成功被执行，则网页查询并返回数据的时间相比于正常的情况下必然会有延迟，不过这同时也依赖网络环境的状态，相同具有使网页返回消息延迟的函数还有 BENCHMARK。通过 Sleep 函数的执行，可以配合使用 AND 语法加入一些条件判断的语句，从而进行简单的正误判断，从而逐位爆破出数据的信息，从而造成 SQL 注入的漏洞。

对于时间型 SQL 注入漏洞的检测，我们可以同时发送一个正常的数据包和一个加入了 sleep 参数命令的数据包，记录两者返回时间，做一个差值，这里我们还可以做些优化来减少网络状态对结果的影响，通过再发送一个较长延迟的数据包，记录返回时间，两次差值取商，如果得到的结果大于某个阈值，则可以判断存在时间型 SQL 注入漏洞。

2.3.4 XPath 注入漏洞

XPath 注入漏洞的原理和 SQL 注入类似，都是由于用户输入的不可信导致的信息暴露问题。SQL 注入是利用了 SQL 数据库的一些语法特性，而 XPath 注入则是利用了 XPath 解析器的一些特性而导致的漏洞。他可以在含有 XML 结构的 URL 网页或者其他信息上加入恶意构造的 XPath 语法代码，查询一些消息或者直接获得更高的信息读取和访问的权限。可以从 OWASP 的报告中看出，XPath 也是最近今年才逐渐被人们重视到的一种漏洞，其出现的原因也和新型的 Web 应用服务方式密不可分。XPath 漏洞可以造成的危害也是不确定的，有时候只能从 XML 文档中读出一些网页有关的配置信息，有时候却能直接拿到网站服务的密码或者口令，直接获得 Web 服务更高的权限。所以其危害也不容忽视。

XPath 漏洞检测也与报错型的 SQL 注入类似，通过用户在 XML 查询语句中输入敏感字符，如引号，XML 注释语法等，来造成 XML 报错，通过匹配这些错误信息来完成漏洞的检测。

2.3.5 XSS 漏洞

XSS 全称为 Cross Site Scripting，也称作跨站脚本攻击。该漏洞发生在用户端，是指在渲染过程中发生了不在预期过程中的 JavaScript 代码执行。XSS 通常被用于获取 Cookie、以受攻击者的身份进行操作等行为[16] 。

针对普通 XSS 漏洞的检测也较为简单，则是在发送的 payload 请求中加入一些待渲染的 JavaScript 代码，然后从服务器的返回包中检测渲染结果，如果该 javascript 代码成功被执行，那么在页面中的元素呈现并不会出现被编码或是一些畸变，若元素还可以在被返回页面中被匹配到未做修改，则说明存在 XSS 漏洞[21] 。

2.4 本章小结

本章主要介绍了现有的漏洞扫描技术的概况和分类，也是现在主流的扫描器在使用的一些扫描思路，实现方法各有不同，但是原理大致相似。

第一部分介绍了基于主机层面的漏洞扫描技术，该技术现如今也被作为内网资产监控，快速安全响应而被使用。在各式各样的互联网服务急剧增加的今天，其后也有无数硬件设备的支持，对于这些设备的安全和维护也十分重要。

第二部分主要介绍了 Web 应用层面的安全漏洞扫描技术，主要从主动和被动两个安全层面来讲，也可以理解为自动化和半自动的区别。这两者也没有孰高孰低之分，都有各自的优缺点，全自动化的扫描节省了大量的人力，只会消耗硬件资源，检测范围广泛全面，可以确保一些基础的安全问题被查出，保证了 Web 服务的基本安全。而半自动化的扫描方式，更深受高水平渗透测试者的喜欢，对于他们来说被动式扫描器更像是渗透测试中的工具一般，对于像金融、科研核心部门、政府机构、大型企业等的网络服务，面对的网络攻击形势更加严峻，对于安全的要求也更高，这时候就需要更深层次的渗透测试来发现更深的安全风险，这时候高水平的渗透测试人员配合被动式扫描器，既能在短时间内完成测试，又能完成更加深入精确的测试。

第三部分主要详细介绍了三大类漏洞的形成原因和检测方式，这些漏洞也正是本文扫描器设计与实现中涉及到的三类漏洞。这也是为后文设计实现扫描器做一个简要的铺垫。

3 被动式漏洞扫描器设计方案

通过上一章节对于现有漏洞扫描技术的概述，我们在本章引出被动式漏洞扫描的方案。可以看出主流的扫描器还是普遍采用了全自动的扫描方式，因为这种方式大大节省了人力资源的消耗，懂得基础的运维技术人员就可以部署和使用扫描器，解决大部分安全隐患。但是对于一些安全性需求很高的地方，这远远是不够的。安全没有足够与不足够之分，真正的安全从来都是无感知的，如果真的因为安全问题引发了严重后果，之前做的安全工作做的再周密都等于 0，如同二进制一般，只要 0 和 1 的区别。所以努力的在安全测试环境加大深度，提高精度，任重而道远。

被动式扫描可以大大提升渗透测试人员的效率，试想对于一个系统，可能存在漏洞的点就有很多，而每一个点又会有很多的测试方法，单靠人力远远无法做到全面深入，半自动化的被动式扫描在这时候显得就尤为重要[20] 。

3.1 需求分析

一个部署在 B/S 端的扫描器，除了完成其核心的扫描功能，同时还要考虑扫描器本身的部署，使用何种 Web 服务，如何部署在服务端，实现完全浏览器即可使用的扫描器。除此之外还需考虑到，本文提出的是基于流量监听的被动式扫描器的实习，如何抓取到测试人员的访问测试流量，也是很核心的一点，立足于这些要求，提出以下论述的需求[18] ：

作为一个 B/S 架构的漏洞扫描器，基本的首先需要部署一个 Web 服务确保扫描器前后端可正常关联，其次就是扫描器的各项功能。

（1）认证登陆功能，因为考虑到非本地部署，所以需要一个登陆验证过程，包括使用 session 校验登陆状态等、确保使用扫描器的自身安全，不会被网络上的其他人任意修改配置和使用。

（2）高并发的数据处理功能，对于网站的流量的全部监听需要在毫秒级完成数十甚至数百 url 请求和转发，才能保证代理的正常运转。使用 python 的 tornado 框架，实现高并发请求的快速响应。

（3）流量监听功能，这一功能是设计被动扫描器的核心功能，需要对测试人员的流量进行监听并抓取，保证扫描器能够接收到流量包，并进行相应的操作。一般来说对于 Web 服务需要抓取的只有 HTTP/HTTPS 的流量即可，可通过浏览器设计代理从而实现。

（4）漏洞扫描功能，这一功能也是扫描器的核心功能，其目的也是为了

从流量中发现漏洞，改功能需要配合规则模块来使用，包括流量的解析，payload 重放，返回包接收等等。

（5）扫描规则模块，该部分主要负责收集和整理针对于不同漏洞的扫描规则，配合漏洞扫描模块使用，能够达到漏洞扫描模块更具不同等级的规则来进行不同级别的扫描，提高扫描精度，或者为了扫描时间放弃一定的扫描力度。

（6）完成基本前端页面的设计，结果可视化的需求，扫描器部署在服务器之后可以长期运行，使用时只需通过浏览器登陆，然后开启代理监听功能即可，所以需要把所有的扫描控制接口都写到前端，以达到使用前端即可开启扫描器、配置扫描器、对扫描器进行其他操作等，之后对于扫描结果的查看，具体发送的请求包和接受包的详情查看。

（7）数据库工作队列需求，对于抓取到的流量生成待扫描的任务、正在进行的扫描任务，完成的扫描任务及结果，都需要持久化的存在，即可以需要维护一个工作队列，完成任务的建立，移动到不同的状态下，最后可在完成的任务里面看到任务详情，其次为了保证 Redis 队列的稳定运行，还需要稳定保障机制，当 Redis 因意外退出或者无响应时，可以重启 redis 服务。

（8）配置 API 功能，对于扫描器的配置，全部可以移到前端，只需要通过前端就可以修改扫描器的配置文件，和规则配置文件，无需关系扫描器在服务器上的运行。

3.2 被动式扫描器的设计与架构

基于以上提出的这些需求，此部分主要详细论述扫描器的整体设计思路、设计流程、架构设计和部分核心模块的设计思路。

3.2.1 扫描器的整体工作流程

扫描器首先需要抓取到测试人员的流量，来存储在扫描器的数据库中，从而形成一个扫描任务，在抓取流量过程中，扫描器的工作是开启监听端口并转发，也就说只是实现一个在线代理的功能，并不影响测试中的正常访问，可以让测试者进行持续的网页浏览测试。抓取到足够的流量之后，也就是任务队列有足够的任务累计之后，可以选择开启扫描功能，此时扫描器就会从准备队列中取出任务开始扫描，这个过程中会使用核心的扫描功能对所有待测试任务进行测试，然后将已完成的任务移至扫描完成队列，供以测试者查看。除此之外，扫描器还可以同时开始流量监听和扫描功能，一旦有流量被

抓取到了准备队列中形成任务就开始扫描。对于核心的扫描功能，首先是对于已经抓取到的流量进行解析，分析出需要请求的 url 和其他有效字段，然后通过调用相应的漏洞扫描规则，在请求包中加入合适的 payload，来形成新的 request 包发送出去，将此时收到的 response 包进行内容匹配，如果在 response 包中有漏洞规则被匹配到，则说明存在相应漏洞， 并将结果可视化在前端。

扫描器工作流程如图 3-1 所示：

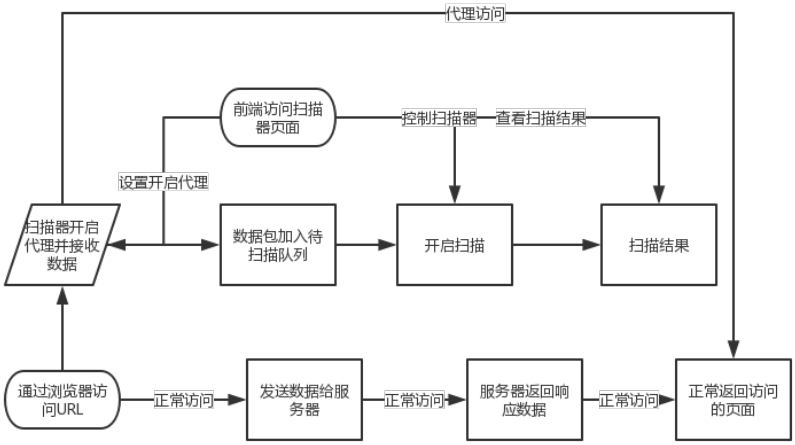

图 3-1 扫描器工作流程图

3.2.2 扫描器整体架构设计

为了满足流量监听需要快速高性能响应的需求，所以选择了 tornado 作为 web server 的底层框架，该框架的核心便是非阻塞式异步响应，它在底层就通过基于 epoll 的单线程异步架构来构建 web server 服务，使得他具有优秀的大并发处理能力。

大部分 Web 应用都是阻塞性质的，也就是说当一个请求被处理时，这个进程就会被挂起直至请求完成，多数情况下会使用多线程技术来解决这个问题。使用多线程池，每当服务器接收到一个客户端的请求就立刻分配一个新的线程来响应。如果线程池中设置了最大可新建 20 个线程，那么服务器的处理性能也就相较于单线程处理方式提高了 20 倍。但这仍然不够，对于浏览器的毫米级别的响应来说，这还是太久了。虽然可以通过无限提高线程数来提高响应速度，但是这个开销最终是得不偿失的，且这种提高线程数量的方式也永远不可能将并发提升到每秒 100 个请求处理的的效率，资源反而会消耗在大量的任务切换开销中。但是如果使用异步 IO（asynchronous）的方法实

现，那么达到每秒成百上千个请求处理的的速度还是非常容易的。在 tornado 框架中，tornado.ioloop 就是 tornado web server 异步最底层的实现体现。ioloop 的实现基于 epoll，而 epoll 是 Linux 内核为处理大批量文件描述符而作了改进的 poll。通过轮询 socket 连接的写入状态， 来达到同时进行多个连接的效果。Tornado 的异步原理图解如图 3-2 所示。这种处理性能，也正好可以满足我们流量检测、在线代理的功能需求。完美解决测试人员无卡顿访问网页，同时还可以记录响应流量。

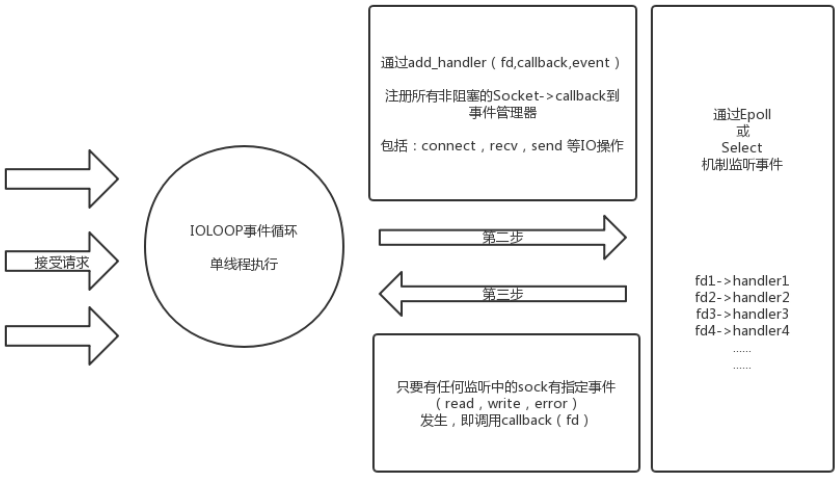

图 3-2 Tornado 异步原理

除此之外，为了持续的完成任务，还设计了 redis 任务队列，通过维护不同的 redis 队列来使得任务跟踪明确，任务完成状态清晰化。测试人员可以选择一次性进行流量监听和抓取，等到积攒到一定的任务数目之后，再进行统一的扫描，亦或者在进行测试的时候就开启扫描，一旦有任务建立，即可进入扫描队列，最终所有的任务都将进入完成队列，对于有漏洞检出的任务，进行特出标记和统计，这样测试人员就可以知道哪些被测试的连接或者接口存在漏洞和风险问题。

3.2.3 扫描器的漏洞规则匹配的设计

为了使得扫描器具有可扩充性，方便随时增加扫描器的功能和漏洞扫描范围，通过编写不同的 XML 文件来实现对于不同漏洞的规则设计，每一个 XML 结构树即针对某一类漏洞进行设计，在 XML 中需要定义对于漏洞测试的 request 请求内容，一般就是漏洞测试所需的一些 FUZZ 结果的收集和整

理。在此之前如果一种类型的漏洞有 30 种的测试方式，那么手动测试的时候，测人员就需要对同一个可能存在漏洞的 url 进行 30 次不同的手工测试，或者临时编写脚本或者使用已有的工具进行 30 次测试，无论哪一种都是十分耗费资源人力的，而使用 XML 文件定义，测试人员只需要完善或者稍微修改已有的规则文件即可进行测试，扫描器在后端自动对应不同每一条规则发出 request 包，并接收响应进行匹配检测。无疑节省了大量的时间和精力。

3.3 本章小结

本章主要通过第二章的综述，来引出本文核心即被动式的 Web 漏洞扫描器设计。首先通过已经了解和研究过的方法，来对扫描器应该实现哪些功能提出了详细的需求分析，这也是一个好的工程开始的第一步。有了明确的需求分析之后，在第二部分基于提出的这些需求，从三个方面来详细讲述了扫描器的设计思路。分别是扫描器整体的设计思想及工作流程、核心架构的选择和核心功能模块的设计思路。

4 被动式漏洞扫描器实现

本章主要根据上一章节的设计方案，进行详细的编程实现，在编程实现的过程中对于不同功能模块使用不同的实现方法和实现策略，保证各个功能模块之间的功能对接流畅，同时对于设计中的一些不足进行改进和调整。通过 python3 的编程，真正实现该扫描器的核心功能和基本 Web 界面，以便进行下一步的测试和分析。

4.1 Web Handlers 模块的实现

该模块是整个扫描器运行的基础，其他所以模块也是依托于该模块进行开发，该模块主要实现了从 Web 界面进行各种功能触发，也是在扫描器使用过程中，测试人员可以看到并且操作的界面，它的设计和是实现决定了扫描器的易用性，同时一些页面的小功能，也可以及时的在前端解决一些扫描中遇到的问题和意外，例如任务队列的清空，扫描器状态的重置和复原。

其详细功能主要包括扫描器登陆验证、开启扫描、开启流量监听（代理）、配置扫描器各项属性、将各种扫描器指令通过可视化方式实现在 Web 前端，并且进行一定的 UI 设计，保证扫描器在使用过程中的美观和简洁易用性，例如对于扫描结果的状态使用不同的颜色标注，查看任务详细信息的时候，是一个阅读性较强的界面，而不是单纯的数据包参数。

在编程实现中，该模块分为两部分实现，首先定义路由，对于不同的 URL 和操作进行路由设计，Web 端通过 GET 方式向后端扫描器设计提交数据，并且对于不同的提交路由，进而由不同的函数进行处理。该模块的实现主要定义在程序 Web 结构下的 main.py 中的函数，这些函数主要有：main.IndexHandler 、 main.LoginHandler 、 main.LogoutHandler 、 main.PageNotFoundHandler、main.ResetScanHandler 等。

其次是一些静态文件的设计，包括 HTML、CSS 的一些文件，目的就是为了完成界面可视化。该模块实现之后就有了一个 B/S 架构的扫描器的雏型，在 Web 端和扫描器后端的交互基本完成之后，剩余的就是扫描器的其他核心功能的实现。

4.2 Proxy 模块实现

该模块也是扫描器的核心模块之一，基于流量监听的被动式扫描器，流量监听可谓是很重要的的一个环节了，它涉及到流量的抓取，在对 Web 应用

层的漏洞进行分析，大多数还是以 http/https 协议为主的，所以这里我们通过使用 python 写一个在线 socket 代理（proxy）的服务，来完成我们的流量监听功能。

其基本原理就是，首先开启一个本地端口，创建一个 socket 连接通道进行监听，当测试人员在浏览器中设置了代理方法，即可让浏览器访问的流量完全转发到代理服务器，也就是我们监听的端口，然后在监听的端口进行循环读取，一直读完整个 http/https 的请求包，在读的过程过也是启用 threading 库，进行多线程任务，这样可以很好的快速接收来自不同的请求的数据包，在接收数据包的同时也对不同的请求包进行判断，是 GET 请求还是 POST 请求，除了请求的 URL 以外，还附加了什么参数，这些都可以通过解析数据包来得到，得到了这些我们就可以把一个数据包按照我们想要的格式，规整的存在 Redis 数据库中，形成一个扫描任务，若此时扫描器的扫描状态是打开的话，任务会立刻被加入到正在扫描的队列进行测试。

解析完数据包建立任务之后，并没有完全一个代理服务器的任务，代理服务器就是，替代原有的服务器进行访问，然后还会把请求收到的结果再返回回去，让浏览器的访问返回结果和正常访问看起来没有什么区别。所以我们还需要对这些接收到的 HTTP/HTTPS 数据报文，按照报文中原有的意图去进行请求，将得到的结果再返回到 socket 连接通道中，这样就可以再返回到浏览器中，让测试人员看起来和正常访问并无多大区别。

除此之外，还需注意一个 HTTPS 的问题，因为 HTTPS 使用的加密传输协议，并不能像 HTTP 请求包一样，可以直接从 socket 连接中读到明文的内容，所以在我们接收到 socket 中的数据时，先要进行判断是那种类型的协议，若果是 HTTPS 的协议，在我们测试的过程中，则需本地颁发一个证书，即一个 PRIVATE KEY 和一个 CERTIFICATE，通过这两个密钥和证书，在读取 HTTPS 报文的时候使用 ssl.SSLContext 函数并将证书加载到 HTTPS 的报文中，即可接收到解析过后的 HTTPS 报文，看到里面的详细内容，一般来讲，直接颁发的本地证书对于浏览器来讲并不认可其安全性，会在访问 HTTPS 的时候提示有安全风险，不过可以通过调试模式打开浏览器来忽略这一风险提示，也可以在浏览器中将自己的本地证书设置为可信任证书，并使用游览器再次生成的证书即可。

4.3 漏洞扫描模块实现

该模块也是扫描器的核心功能，当我们完成了基本的 Web 处理功能，

Proxy 代理功能，接下来需要考虑的就是如何通过分析从我们得到的这些数据中发现出漏洞。

扫描模块的原理是，先读取扫描器的扫描相关配置，这些配置主要包括各个针对于不同漏洞规则文件是否需要修改，是否开启对于某一类漏洞扫描、扫描器模块同时可以进行几个任务的扫描，扫描深度是多少等。拿到这些配置参数之后，扫描器就知道对于接下来要测试的 URL 需要进行哪些测试流程。然后扫描器从 Redis 任务队列中取出一个等待检测的任务，分析其数据包，根据之前定义的扫描规则，从对应漏洞的 XML 中进行加载 payload 重放，数据包、再接收数据包、进行比对匹配，判断是否发现漏洞，如果有发现漏洞则对一个 message 参数进行标记，以便后续知道该测试项存在漏洞，然后进行前端现实，供给测试人员查看。

扫描模块的实现，主要是在 scan.py 文件中编写，针对对每一个类型的漏洞，我们可以先总结出其公共扫描部分，比如都需要从任务中获取数据报文具体内容进行解析，然后根据规则文件发送请求，最后再接收响应包进行匹配，所以将这些操作写成一个公共函数 common_scan，然后再根据不同的漏洞类型进行具体的操作，每个漏洞类型进行一个单独的函数编写，只需通过读取之前扫描配置文件的参数，看是否开启了的对于这个漏洞的扫描测试，然后调用相应的函数即可。

4.4 规则模块的实现

规则模块的实现依赖的是对于 XML 文件 dom 树结构的解析。XML 是指可扩展标记语言（EXtensible Markup Language），被设计用来传输和存储数据。XML 文档形成了一种树结构，它从"根部"开始，然后扩展到"枝叶"。每一种类型的漏洞都会对应一个不同的 XML 规则文件，在漏洞扫描模块直接被调用。XML 文件 dom 树的一般结构如图 4-1 所示：

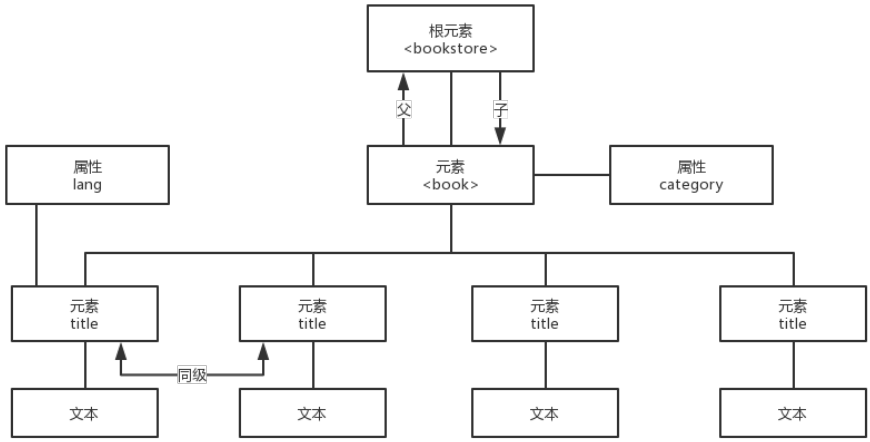

图 4-1 XML 文件 Dom 树结构示意图

其中文档开始都由一个 rules 根元素来包裹，rules 元素下面有属性不同的子元素定义为 couple，couple 元素下又有不同的子元素，这些子元素可以改通过对于不同漏洞来编写，比如说针对 SQL 注入漏洞需要加入 payload 参数的测试，则需要一个 request 元素，来放置那些需要在测试的时候加入数据包的参数，然后还需要一个 response 元素，来放置到时候需要匹配的特征，同一个漏洞在不同的服务上可能提示不一样，我们在这里尽可能全面的收集这些特征。这个子元素不是一定的，在本文实现的扫描器中，针对于 XSS，布尔型 SQL 注入，时间型 SQL 注入，报错型 SQL 注入和 XPATH 漏洞中都有不同的定义。

其次，通过对于 couple 元素赋予不同的 id 属性，来在扫描的时候决定扫描的深度，如果在扫描配置中配置了扫描深度为 3，则会调用 rules 下面的三个 couple 子元素，并且属性 id 分别等于 1、2、3 的进行测试，不过这也意味着需要消耗更多的时间和资源。有时候为了快速测试，只需要调用第一级别的规则即可。分级编写规则的目的也是为了能够更加全面的描述漏洞的规则，对于一个漏洞的描述可能会有几十条几百条特征，但是我们最常见的或许也就只有十几条，所以在 XML 文档中对于漏洞的规则分级描述是十分有必要的。

4.5 Redisopt 模块实现

该模块的实现主要是为了扫描器可以方便的联动 Redis 数据库，同时也是为了集成统一对于 Redis 数据库的操作，将连接数据库、添加内容至数据

库、取出数据库内容，对数据库中的数据进行编码解码处理等进行函数统一实现，并单独写在 redisopt.py 模块中，避免在代码实现过程中反复在每个文件中重复数据库相关的操作，降低代码重复率，只需要调用该模块的函数即可。

同时在具体的实现过程中，对于加入工作队列的任务进行编号，编号的实现是通过对于接收到的数据报文取一个摘要 hash，这样可以最大限度的避免任务重复，因为有时候对于同一个 URL 可能会并发好几个请求，但是他们是不同类型的请求方式，数据报文中的参数也不尽相同。而且在数据报文加入数据库和取出数据库的时候都会进行一个 Base64 的编码处理，这样处理的目的是为了避免一些测试中的敏感测试参数对数据库产生影响，同时也可以避免乱码问题，增加了数据库存储的稳定性。

4.6 配置信息 API 模块实现

该模块是为了集成一些扫描器的基本配置项，以供测试人员可以直接在前端对一些扫描器的参数进行修改。这些配置包括

（1） Redis 相关配置

测试人员可以决定使用部署在任何地方的 Redis 服务器，一般来讲 redis 服务和扫描器的后端服务会部署在同一个机器上，但是一些情况下，比如扫描任务多，存储量过大，或者需要分布式的接入其他测试人员的数据库等，就可以在此处配置 Redis 服务的地址、端口号、登陆密码。

（2） 扫描器登陆账户相关配置

之前的 Web Handler 模块中就提到了扫描器自身安全的问题，为了防止除了测试人员以外的用户访问扫描器，造成不必要的损失，在 Web 端需要通过验证登陆扫描器，再此处即可定义登陆的用户名和密码，以及加密 cookie 的 secret key。不过并不建议在生产环境中使用这种配置方法，如果没有详细的修改密码的逻辑，直接显示原密码和轻易可修改密码是不安全的。

（3） 扫描器 Session 相关配置

Session 的配置也是为了配合登陆登陆而存在的，为了方便测试人员的使用，一次登陆之后服务器会分发一个 Session 给浏览器，一段时间内，再次访问扫描的 web 服务并不会要求登陆，在这里我们可以自由的配置 Session 的大小和 Session 过期的时间，太长时间的 Session 也并不安全，在此处可自行调节。

（4） 扫描器部署访问的配置

该部分主要是配置扫描器部署启动时所指定的 IP 地址和端口，若扫描器的

部署和测试人员的使用并不在同一台机器上，为了可以使得扫描器可用，需要将部署时的 IP 修改为 0.0.0.0，这样才能通过另一台机器访问到存在于同一网络下的扫描器。端口默认为 8000，一般在本机测试时，访问的默认为 127.0.0.1:8000 即可看到扫描器登陆界面。通过前端改写该项配置，可以灵活解决端口占用，无法访问等问题。

（5） 扫描任务相关配置

这个配置选择主要是为了定义扫描器的扫描性能，取决于部署扫描器的机器性能如何，对于内存较大，cpu 处理能力较强的机器，可以上调扫描的最大线程数和扫描级别，达到更加快速，更加全面的扫描效果。同时该部分还可以配置是否匹配多个规则，有一些 URL 在测试的时候很有可能可以同时符合多规则的匹配，我们在此处可以选择，是否在第一次检测到漏洞存在之后还会用其他的规则继续测试该 URL。继续测试可以挖掘出该漏洞的更多的信息，而选择只匹配一次则可以更加节省扫描的时间，这个可以是情况而定，是需要更加快速的结果还是需要更详尽的测试结果[22] 。

（6） 黑白名单配置

黑白名单主要是为了方便测试者灵活的使用扫描器，比如测试者需要一边访问被测试的网站，同时还需要 Google 搜索一些问题，而 google 并不是我们想要检测的目标，这时候我们可以将 google.com 这个域名加入到扫描黑名单中，这样就可以避免扫到我们不需要的网站和流量。同时对于一些 URL 其实并不是我们访问的网站，只是网站上加载的静态或者动态资源，或者一些使用的插件等产生的流量，我们可以通过将这些常见网络资源的扩展名加入黑名单，这样我们也不必去关心它们。默认情况下，扫描器会接收所有的来自浏览器代理的流量，通过黑白名单机制可以灵活的控制测试范围。

4.7 本章小结

本章节主要从六个核心模块，来论述按照上文设计要求所完成的基于流量监听的被动式漏洞扫描器的实现细节，扫描器的编写主要是通过 python3 语言来完成的，其中的特征规则模块是使用了 XML 语法来实现。通过这六个核心模块实现，扫描器已经可以初步运行。除了基本的编程实现，还进行分别的单元测试，确保每个模块的独立功能正常且完善。以及 Redis 服务器的安装部署和初始化配置，确保了其在扫描器中的可用性。

本章中说明的都是核心模块和技术的实现细节，还有包括前端 CSS、HTML、Javascript 等的编写，一些系统模块之间的联通功能的实现没有提及，

对于前端内容的实现为了美观，借鉴了一部分已有其他开源扫描器的外观，加快了编程实现的进度。完成扫描器的相关核心功能的实现，使得扫描器可以正常运行，就可以进行对于扫描系统的详细测试和结果分析。

5 系统的测试验证与分析

5.1 测试相关环境

（1） 扫描器系统部署环境：

Mac OS X 10.14.4 18E226

处理器：3.1 GHz Intel Core i5

内存：16 GB 2133 MHz LPDDR3

（2） 测试对象部署环境

虚拟机中运行的 64 位 windows 10 操作系统

5.2 测试对象

5.2.1 DVWA 和 SQL-lab

在测试过程中我们为了验证扫描器的可用性，选择了 DVWA 和 SQL-lab 这两个开源的漏洞环境进行测试。

DVWA (Damn Vulnerable Web Application) 是一个包含了各种常见的 Web 应用漏洞的 Web 应用程序。它是基于 PHP/Mysql 搭建，其主要目的就是为了帮助安全研究人员和安全测试人员快速的搭建漏洞环境进行测试和学习。DVWA 中包括了我们需要测试的 SQL 注入漏洞和 XSS 漏洞，可以方便的用来检测扫描器的效果。

而 Sqli-labs 是最出名的用来学习 SQL 注入技术的漏洞环境，最初由一位印度程序员编写。这个环境中包含了六十多种针对于不同类型 SQL 注入的环境，非常适合我们测试之前提及的本文中扫描器设计实现的三种 SQL 注入漏洞。即报错型 SQL 注入、时间型 SQL 注入、布尔型 SQL 注入。

5.2.2 测试对象的部署

对于这两个漏洞环境的搭建，都是采用了虚拟机中部署运行的方案来实现的，在虚拟机中安装 phpstudy，然后分别部署 DVWA 和 SQLlab 的环境，虚拟机网卡选择 host-only，通过本机来访问这些漏洞环境进行测试。

（1）虚拟机开启 host-only 后分配到 IP：10.37.129.3，然后从官网下载 DVWA 的程序源码，复制到 phpstudy 的 www 目录下，进行简单配置，即可成功访问 dvwa 环境页面。

访问 http://10.37.129.3/dvwa/index.php 结果如图 5-1 所示，可以看到已经可以成功访问 dvwa 的首页并成功登陆进去，图片右侧所展示的就是不同类别的漏洞目录，点击不同的漏洞类型，进入相应的页面就可以进行测试。

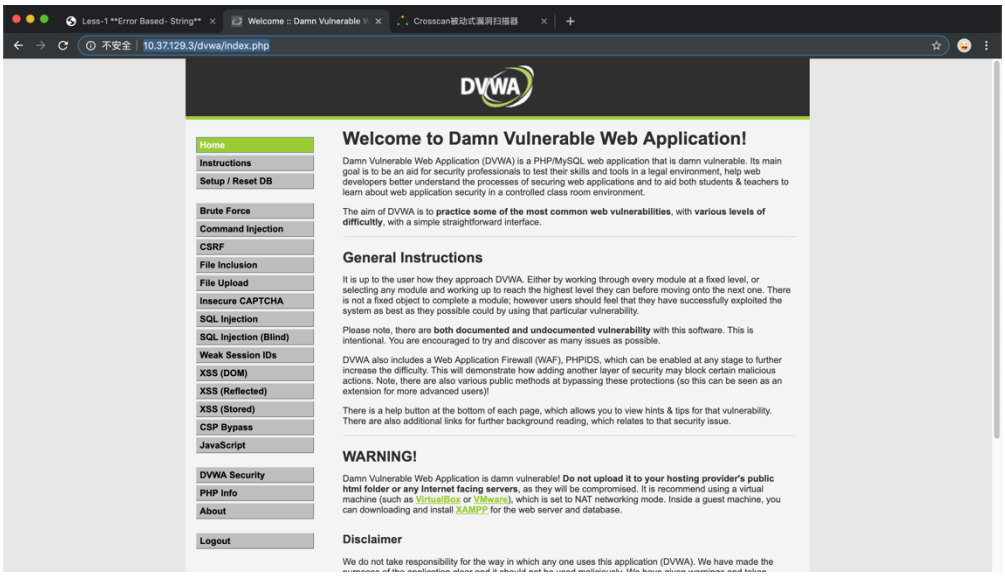

图 5-1 成功部署 dvwa 测试环境

（2）下载 SQL-lab 的源码到 phpstudy 的 www 目录下同样的进行配置，然后访问 http://10.37.129.3/sqli-labs/，结果如图 5-2 所示，我们可以看 SQLi-LABS PAGE-1（Basic Challenges）的字眼，这表示我们当前所处的漏洞环境的大类级别，对于具体的环境，我们点击在下方 less-1、less-2 的字样进入，然后开始相关的测试。

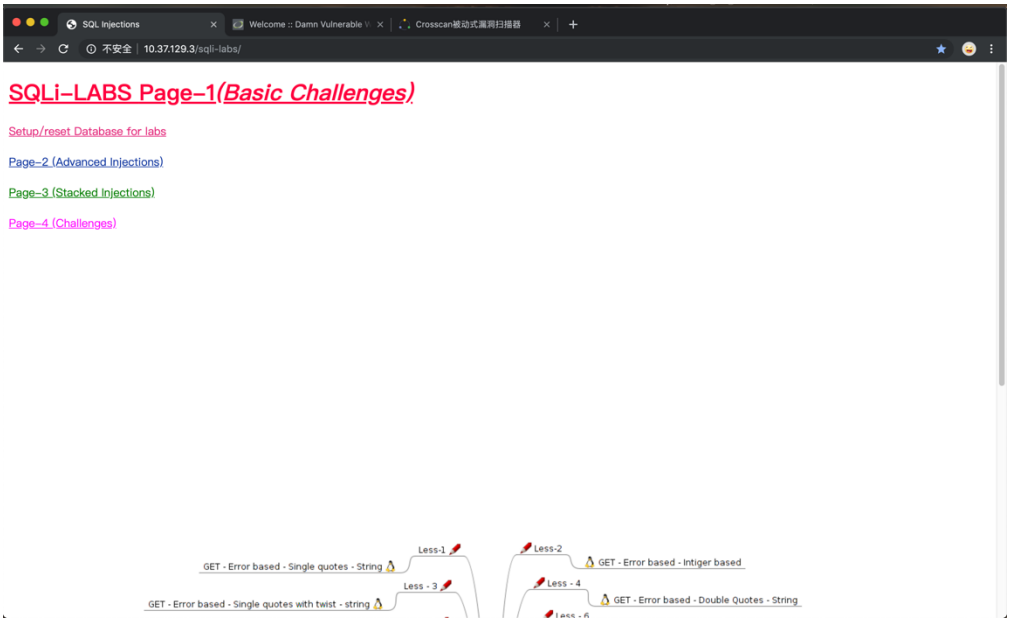

图 5-2 成功部署 SQL-lab 测试环境

5.3 测试过程

（1） 开启扫描器

扫描器的运行需要两步操作，根据已经写好的 redis 配置文件启动 redis-server 服务，其次使用 python3 运行扫描器。为了方便使用 python 环境，在本机测试时使用了 python virtualenv 工具控制 python 环境，所以在启动之前先激活一下 python env 环境。

具体操作输入的命令如图 5-3 所示，首先执行的第一条命令是用来激活 python virtualenv 环境的，这样可以使得我们在接下来成功的运行扫描器。第二条命令是用来开启 redis 数据库，开启的时候是根据提前写好的 conf 配置文件来开启的，配置文件中包括了 auth 认证的密码等配置，以便于稍后扫描器启动时可以正常链接 redis 数据库。第三条命令则是使用 python 开启扫描器，执行后即可看到扫描器的 web app 服务已经运行在 8000 端口。

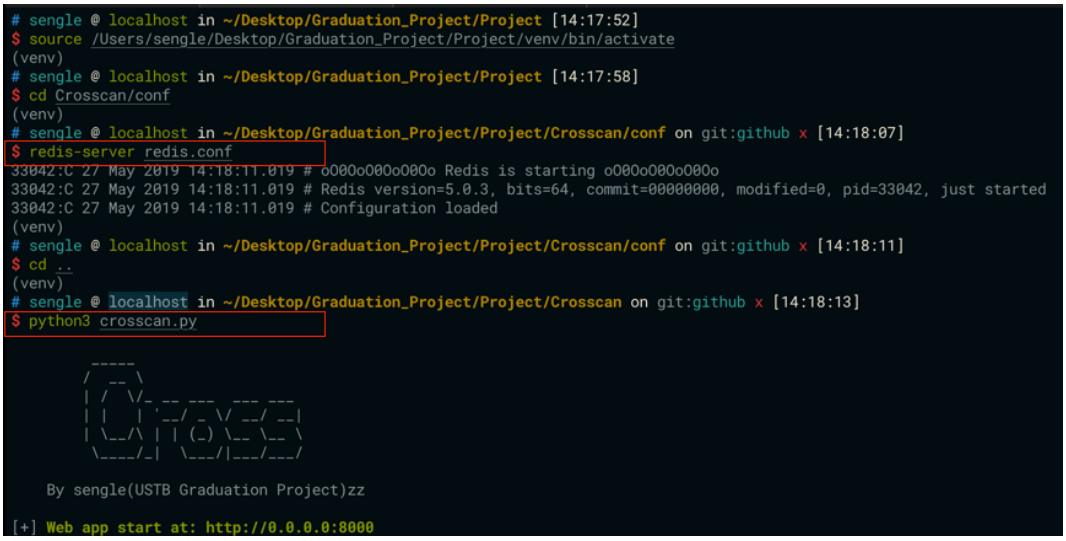

图 5-3 开启扫描器的相关命令

（2） 访问 127.0.0.1:8000 ，登陆扫描器进行初始配置并开启代理，扫描器的前端登陆界面以及登陆成功后的扫描器布局界面，如图 5-4 和图 5-5 所示：

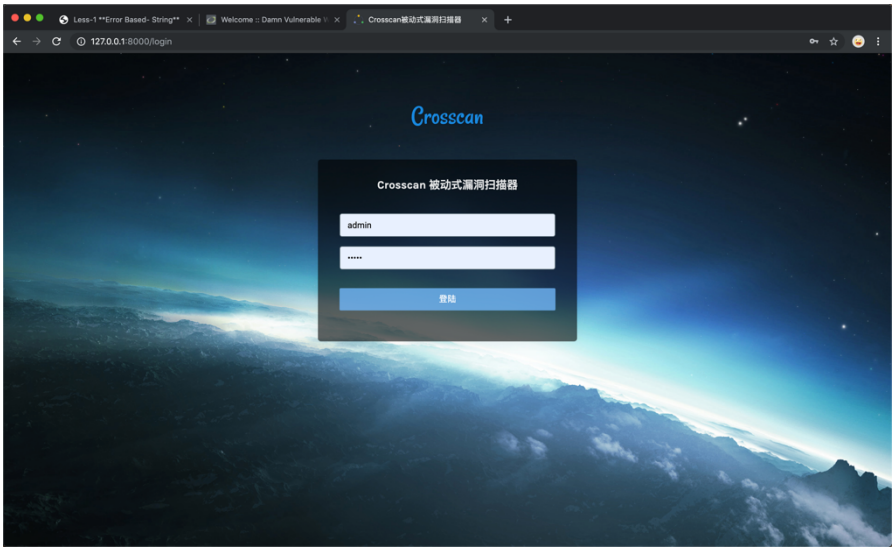

图 5-4 扫描器登陆界面

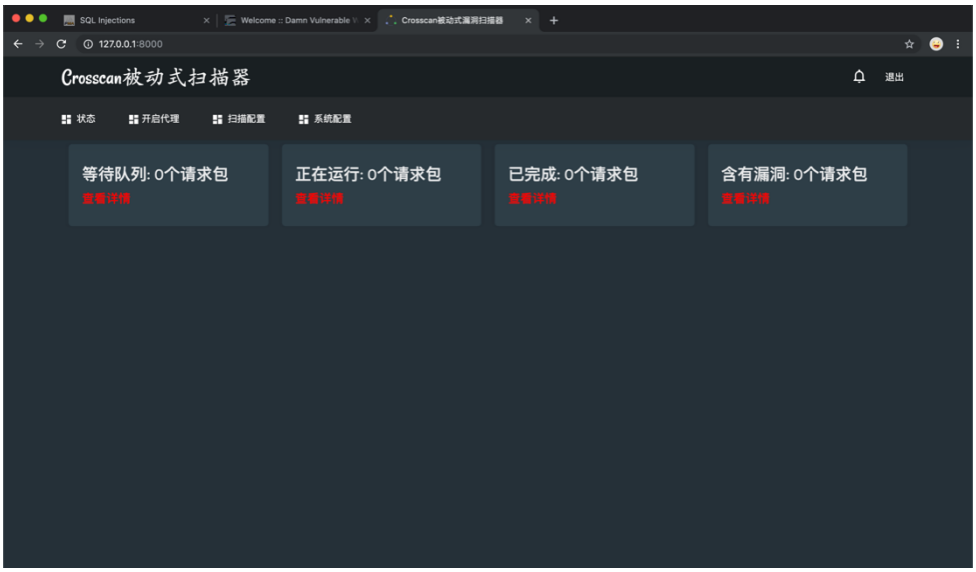

图 5-5 登入扫描器后界面

登陆成功之后，可以看到上方有四个基本选项栏，分别为状态（即当前页）、开启代理、扫描配置、系统配置。下方则是四个队列的基本状况，在扫描器还未使用的情况下，可以看到四个队列中都为空。

点击开启代理，进入代理配置界面，如图 5-6 所示：

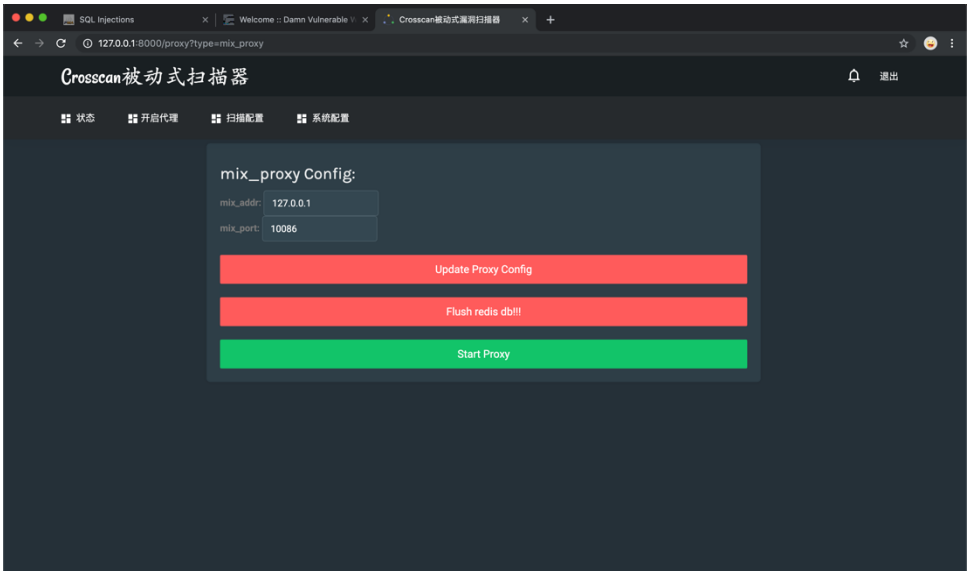

图 5-6 流量监听代理配置界面

对需要开启代理的 ip 和端口号进行配置，因为在本机进行测试所以 ip 填入 127.0.0.1，端口定义为 10086 即可。然后点击 update proxy config 使得配置生效，然后再点击 start proxy 按钮即可开启代理，点击后按钮会变成红色，可以知道现在是否是开启代理的状态。

（3）浏览器通过设置代理，访问被测试环境。根扫描器设置的代理信息配置浏览器的代理设置如图 5-7 所示，我们可以看到对于 http/https 协议设置了代理服务器 127.0.0.1，代理的端口和扫描器中设置的一样为 10086。然后应用设置，即可启用浏览器代理。

图 5-7 浏览器设置代理服务器

（4）设置完成之后，任意访问一个 URL，测试扫描器那边确实可以抓到包，然后就可以进行测试访问了，在这里我们混合正常网站、DVWA、sql-lab 环境来访问一段时间，产生足够的流量包进行下一步的分析。

5.4 测试过程及结果分析

通过一段时间的访问我们可以看到流量已经被抓取到了，并且数据包被解析为任务存放在带扫描队列，等待扫描。经过这次测试，我们一共抓取到的流量一共形成了 102 个任务。具体在扫描器展示的方法如图 5-8 所示，可以看到等待队列中的 hash 值展示，每一个 hash 是根据任务的摘要所生成，所以哪怕是对于同一个 url，但是请求了不同的参数，也会被列为不相同的任务。

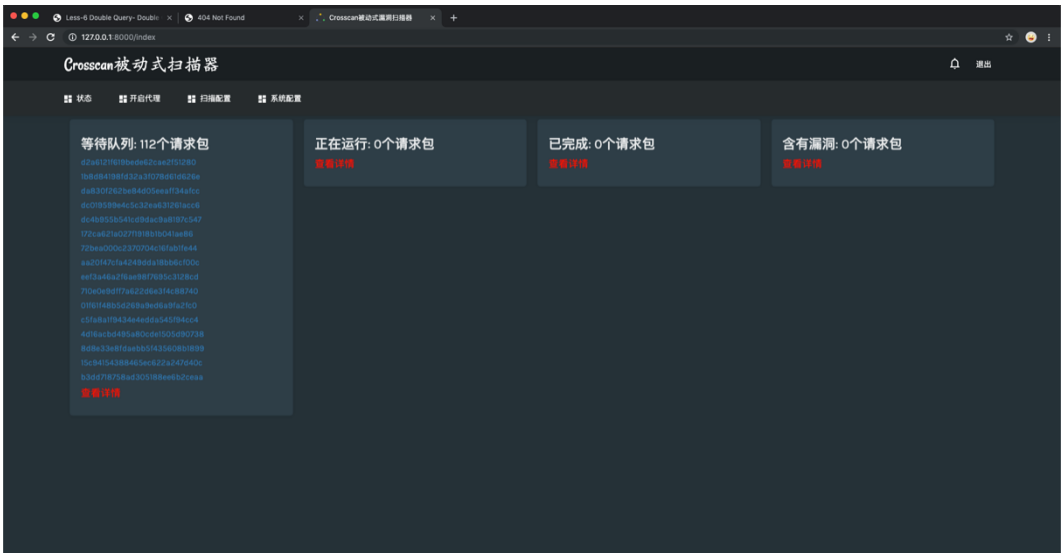

图 5-8 扫描结果形成任务加入等待队列

点击等待队列下方的查看详情，还可以看到每个数据包的详细内容，如图 5-9 所示，在列表界面简单的显示了数据包 hash、请求方式和请求的 url 三个字段，点击去每一个数据包 hash 还可以看到具体的数据包内容，包括请求的 header、参数、状态等。

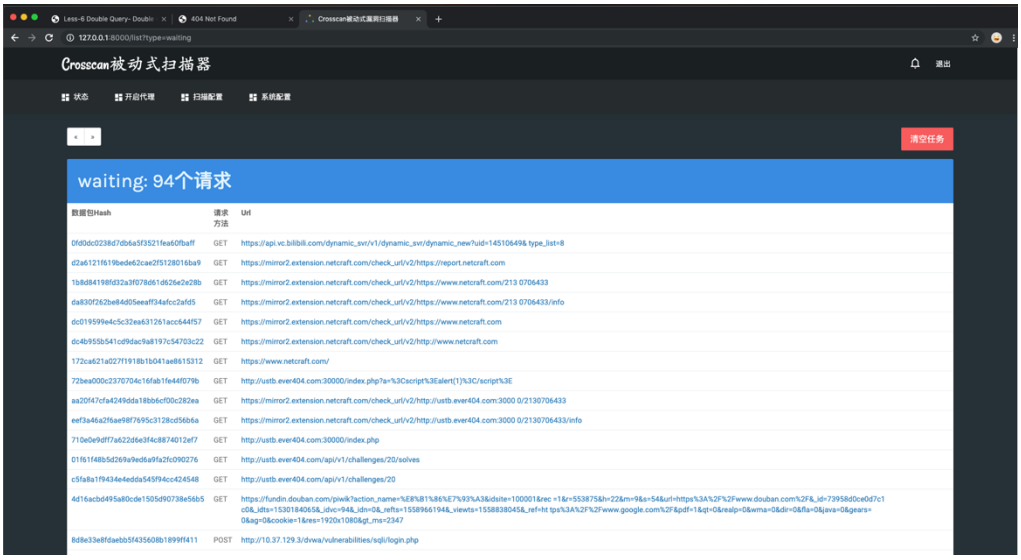

图 5-9 等待队列数据包详细内容

在扫描开启之前可以对整个扫描系统进行相关配置，点击“系统配置”选项，可以对扫描线程数、扫描深度等进行进一步的定义。如图 5-10 所示，我们已经配置了 redis 服务器的 host 地址和端口，由于本机测试，所以都是 host 和 port 都是默认设置，对于账户设置则是因为在本机测试而使用了一个弱口令，如果将扫描器部署在了远程服务器，为了服务器安全则一定要将口令修改，对于 session 配置则是用来控制登陆状态，如果测试环境频繁变动建议设置长度和时间都稍短。

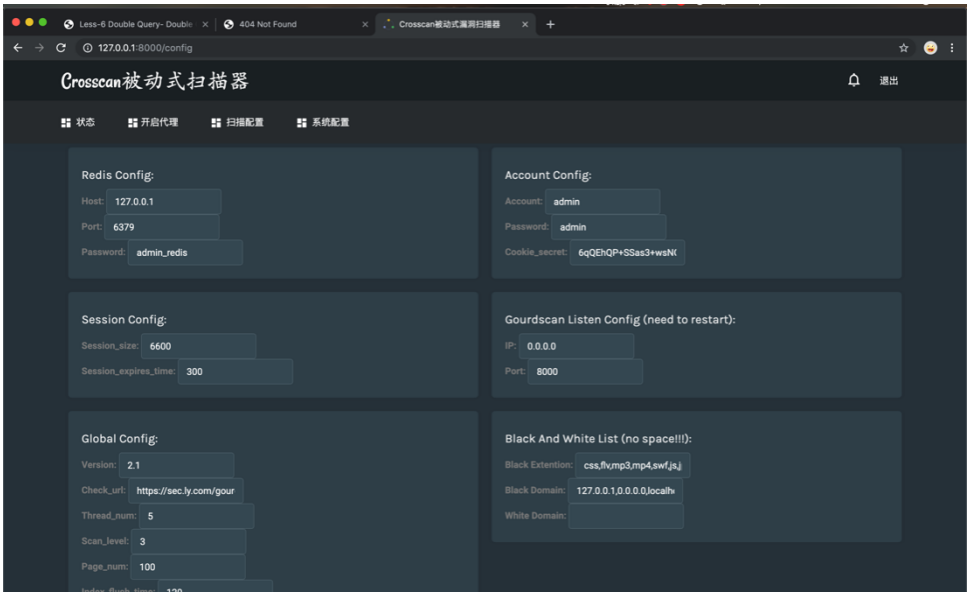

图 5-10 扫描器系统配置模块

接下来就是对这些任务展开扫描，点击四个选项中的“扫描配置”如图 5-

11 所示，我们可以看到最上方有一个开启扫描的按钮，点击之后即可开启扫描状态。

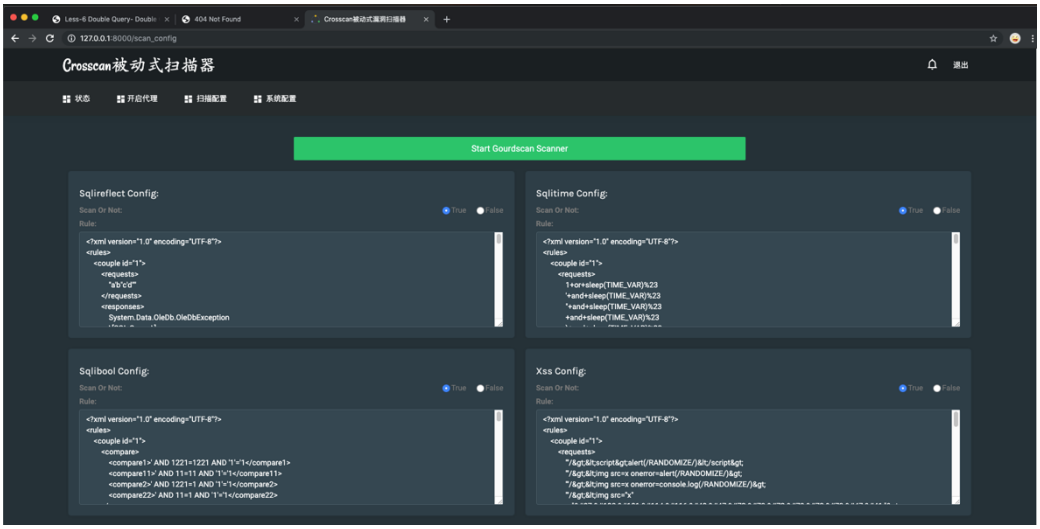

图 5-11 扫描配置界面展示

同时，在这里我们可以看到针对不同漏洞的规则文件的详细内容，也可以在此处直接进行修改，点击绿色按钮开启扫描。

如果在开启扫描的时候没有选择后台运行，我们还可以在命令行中看见有五个任务被加入扫描队列，如图 5-12 所示，可以看到开始的五个任务的 hash 摘要，同时在扫描器前端我们可以看到，这五个任务被移到了，正在扫描队列中。

```
[I 190527 22:16:04 web:2162] 200 GET /scan_stat?stat=True (127.0.0.1) 38.31ms
[I 190527 22:16:04 web:2162] 200 GET /scan_stat?stat=True (127.0.0.1) 11.47ms
[+] start new mission: b'17d13100001915c436a22ef90b50a26d'
[+] start new mission: b'3a688d0250799c0efba5980e934f05da'
[+] start new mission: b'f9e32aa0de89a9d3b75e3b189dac6294'
[+] start new mission: b'1f432f786c6bdc207d2747fc7416ebee'
[+] start new mission: b'c55c6d4c67dfedef169d115e11d0ca45'
```

图 5-12 开启扫描后任务被加入扫描队列

扫描过程中，扫描器状态展示如图 5-13 所示，可以看到，有一部分扫描任务已经完成，有一部分扫描正在进行，由于在全局设置中设置了同时并发的扫描数为 5，所以同一时间内只能有 5 个任务处于正在扫描的状态，这也是由机器的性能决定的。

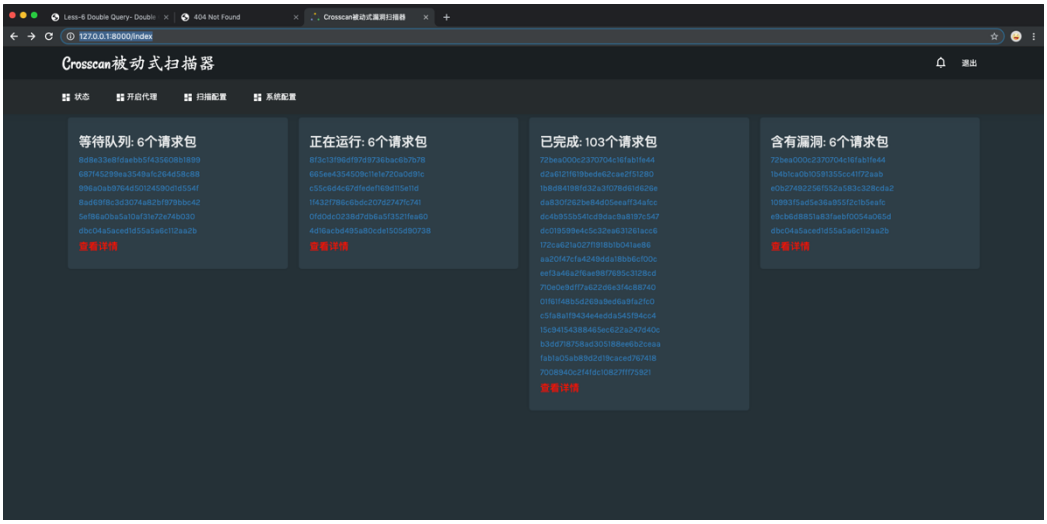

图 5-13 扫描过程中扫描器状态展示

经过一段时间的扫描我们可以看到，任务全部都转移到了已完成队列和含有漏洞队列，从 112 个数据包任务中扫描出了 9 个漏洞，结果展示如图 5-14 所示。

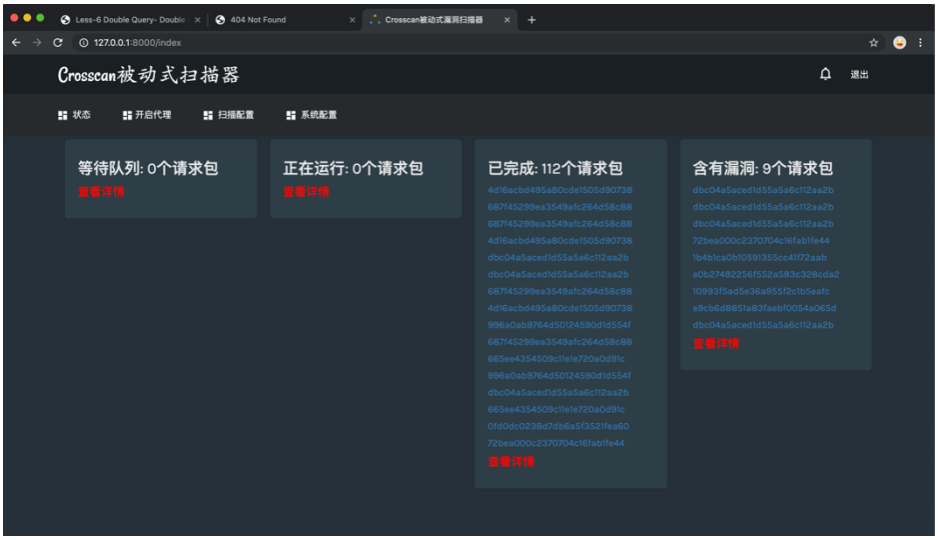

图 5-14 扫描结果展示

点击查看详情我们可以看到对于不同的类型漏洞的扫描结果，同时还会展示扫描过程中匹配到的 payload，便于测试人员进行分析，展示结果如图 5-15、图 5-16、图 5-17、图 5-18，这 4 幅图所示：

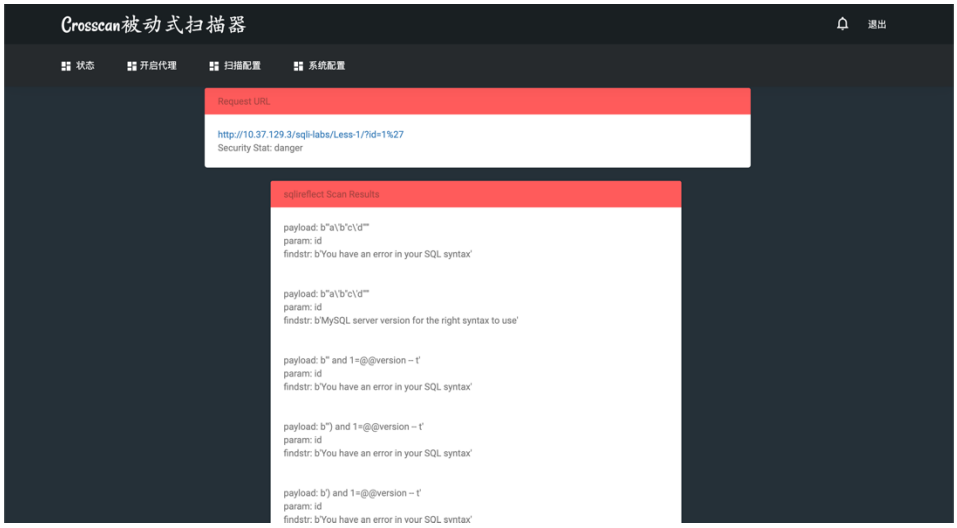

图 5-15 漏洞细节展示 1

图 5-15 展示该 url 存在一个报错型的 sql 注入漏洞，经过测试，我们可以看到注入点为参数 id，而注入的 payload 有很多种都已成功执行，方便测试人员再次手工复现。

图 5-16 漏洞细节展示 2

图 5-16 展示了时间型的 SQL 注入漏洞，相同的也显示出了 payload 和注入点的参数为 id 字段。

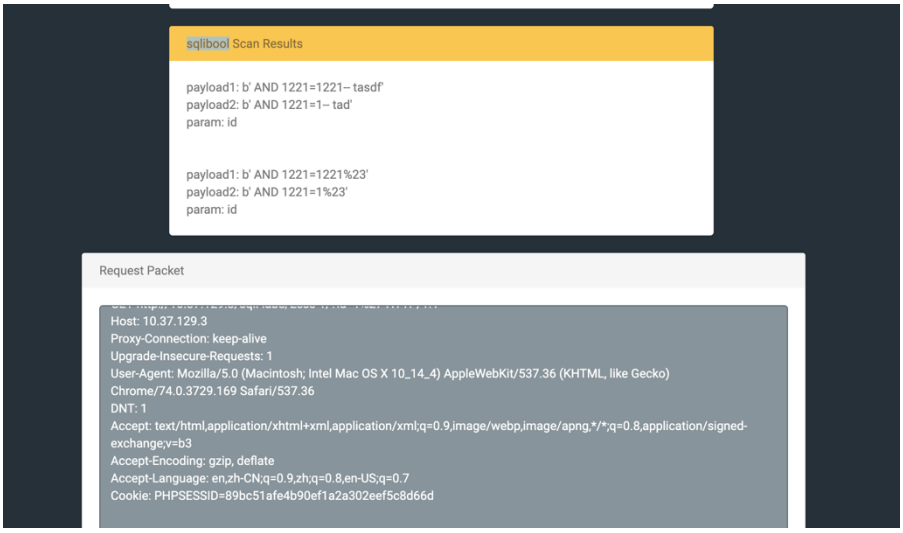

图 5-17 漏洞细节展示 3

图 5-17 展示了存在的布尔型 SQL 注入漏洞，也同样的罗列了成功注入测试的注入点为参数 id 和能够产生不同页面返回结果的 payload1 和 payload2。

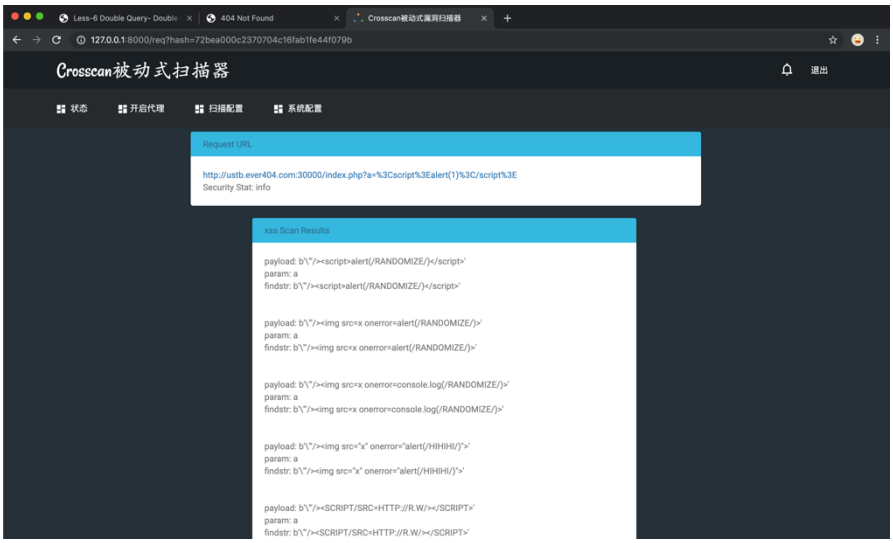

图 5-18 漏洞细节展示 4

图 5-18 展示了 URL 存在 XSS 漏洞，并详细的罗列了可以执行的 JavaScript 语句，对于 payload 和在请求返回包中找到的和 payload 相匹配的字符。由于 XSS 漏洞的危害性理论上是比 SQL 注入的要小，所以安全等级只有 info 级别，同时颜色也使用了较为缓和的蓝色。

通过测试我们可以看到对于报错型 SQL 注入漏洞、布尔型 SQL 注入漏洞、时间型 SQL 注入漏洞、XPATH 注入漏洞、XSS 漏洞有这良好的检出效果。并且可以做到清晰呈现扫描结果，任务展示等。

5.5 本章小结

本章主要从系统测试的环境、测试步骤、测试过程以及结果分析这几个方面来展现描述扫描器工作时的具体使用，通过搭建 dvwa 和 sql-lab 漏洞环境来对扫描器进行测试，从测试的结果可以看出，对于扫描器漏洞规则设计中实现的五种漏洞具有良好的检出型。并且在扫描器的使用过程中可以看出，扫描器的整体设计、界面设计、结果展示设计、配置修改设计都很好的符合之前提出的需求，对于测试人员使用友好，扫描器的设计与实现较为成功。

6 结　论

本章将总结本文设计并实现的被动式扫描器过程中所做的工作和不足之处，并且为进一步的研究指明方向，作出未来的展望。

6.1 工作总结

本文首先对现有的安全形势和背景做了一个基本的调查总结，然后确定了被动式扫描器的研究方向和研究意义，通过查阅相关文献和资料，提出了基于流量监听的被动式漏洞扫描器的设计方案，并设计确定了扫描器所使用的技术架构，基本工作原理和流程，接下来通过 python3 编程实现了需求中提出的功能，开发出了一款可用的被动式漏洞扫描器。

总体来说针对 Web 层面的被动式漏洞扫描器可用性很好，针对本文所涉及的五类漏洞都有良好的检出性，并且对比传统的手工测试，节省了大量的时间和精力。后续只需要维护和研究针对不同漏洞的特征规则，将其收集在规则扫描模块中，即可持续提升扫描器的可用性。对于那些需要深度测试，一般全自动扫描器却达不到要求的目标，被动式扫描器是一个很好的选择。并且安全测试人员也可以将更多的精力集中在漏洞产生和造成影响的核心研究上。

6.2 展望

安全问题是永无止境的。本文实现的被动式漏洞扫描器依然还是很简陋，由于开发精力有限，只能完成针对于五类较为基础简单的漏洞的扫描器实现，但这种半自动化的扫描方式确实深有意义，对于一些 API 的测试，逻辑漏洞的测试，每一个不同漏洞测试都需要有大量的重复性工作来完成， 这无疑是非常消耗精力，如果采用这种半自动型的被动式扫描器，可以更好的代替测试人员完成重复的工作，让他们投入更多的精力到具体的安全目标的发现，漏洞原理研究上。

同时除了扫描器本身之外，作为一款协助人们工作的工具，它也有着很多的发展潜力，比如开箱即用的特性，现如今容器化技术的发展十分迅速，我们可以将扫描器进行封装到容器，无论是何种环境，都可以让扫描器顺利的立即运行起来。跳过了以往的使用 Web 类工具，还需要先搭建所需要的环境的环节。

参考文献

[1] 沈文婷, 丁宜鹏, 郭卫, 等. 计算机网络安全与漏洞扫描技术的应用研究[J]. 科技与创新, 2018(1):154-155.

[2] 耿哲, 王秀美, 王继龙, 等. 基于 Web 的漏洞扫描系统的设计与实现[J]. 计算机与现代化, 2004(11):30-32.

[3] Alkhurafi O B , Alahmad M A . Survey of Web Application Vulnerability Attacks[C]. // International Conference on Advanced Computer Science Applications & Technologies. IEEE, 2016.

[4] Dariusz P, Zachara M , Krzysztof W. Evolutionary Scanner of Web Application Vulnerabilities[C]. // International Conference on Computer Networks. Springer International Publishing, 2016.

[5] Gawron M , Cheng F , Meinel C . PVD: Passive vulnerability detection[C]. // International Conference on Information & Communication Systems. IEEE, 2017.

[6] 梁兴开, 赵泽茂, 黄亮. 基于代理的被动式 Web 漏洞检测研究[J]. 杭州电子科技大学学报, 2011, 31(6):36-39.

[7] 张玉清. 安全扫描技术[M]. 北京：清华大学出版社, 2004.

[8] 吴翰清. 白帽子讲 Web 安全[J]. 信息安全与通信保密, 2015(5):61.

[9] Wu Q, Liu X. Research and design on Web application vulnerability scanning service[C]. // IEEE International Conference on Software Engineering & Service Science. 2014.

[10] 杨新英. 基于网络爬虫的 Web 应用程序漏洞扫描器的研究与实现[D]. 四川:电子科技大学, 2010 .

[11] 张烨青. Web 应用安全漏洞扫描器爬虫技术的改进与实现[D]. 北京:北京邮电大学, 2014.

[12] 牛小菁. 基于数据包分析的被动漏洞扫描技术[D]. 西安:西安电子科技大学, 2015.

[13] 梁兴开, 赵泽茂, 黄亮. 基于代理的被动式 Web 漏洞检测研究[J]. 杭州电子科技大学学报, 2011, 31(6):36-39.

[14] Johari R, Sharma P. A Survey on Web Application Vulnerabilities (SQLIA, XSS) Exploitation and Security Engine for SQL Injection[C]. // International Conference on Communication Systems & Network Technologies. 2012.

[15] Begum A, Hassan M M, Bhuiyan T, et al. RFI and SQLi based local file

inclusion vulnerabilities in web applications of Bangladesh[C]. // International Workshop on Computational Intelligence. 2017.

[16] 李威, 李晓红. Web 应用存储型 XSS 漏洞检测方法及实现[J]. 计算机应用与软件, 2016, 33(1):24-27.

[17] Chen J M, Wu C L. An automated vulnerability scanner for injection attack based on injection point[C]. // Computer Symposium. 2010.

[18] 付堂欢. 基于网络的 web 应用漏洞扫描系统的分析与设计[D]. 北京:北京邮电大学, 2013.

[19] 孟微. 基于 CMS 框架的安全审计技术研究[D]. 西安:西安电子科技大学, 2017.

[20] Erturk E, Rajan A. Web Vulnerability Scanners: A Case Study[J]. 2017.

[21] 李威, 李晓红. Web 应用存储型 XSS 漏洞检测方法及实现[J]. 计算机应用与软件, 2016, 33(1):24-27.

[22] 程诚. 基于模糊测试的 Web 应用安全性检测方法研究[D]. 2016.

[23] Patil S , Marathe N , Padiya P . Design of efficient web vulnerability scanner[C]// International Conference on Inventive Computation Technologies. IEEE, 2017.

在学取得成果

一、 在学期间所获的奖励

2017 北京市级 SRTP 项目结题

2018 年全国大学生信息安全竞赛全国三等奖

2018 年“西普杯”信息安全铁人三项赛第一赛区二等奖

2018 年“蓝帽杯”全国大学生网络安全技能大赛总决赛三等奖

致　谢

大学四年时光匆匆而过，无论留下的是记忆还是遗憾，我们终将进入人生的下一段里程。

毕设期间感谢我的指导老师王志明教授对我的仔细指导，同时也要感谢我的室友，在我需要帮助的时候能够及时伸出援手。最重要的是，感谢我的父母，在我一次次遇到生活中的选择和和心理上的困惑时都可以耐心的疏导我。

大学的时光已经不再，我想再多的遗憾都没有意义了，6 月是一个美好的毕业季，接下来的生活或许有些艰辛、或许存在困顿，但是最重要的是能够不忘初心，坚持的走下去自己选择的路，希望能在多年以后遇到一个美好的自己。